```c
#include <stdio.h>
#include <stdlib.h>

typedef struct
{
    int dato;
    char palabra[30];
    struct nodo2 *anterior;
    struct nodo2 *siguiente;
}nodo2;

typedef struct
{
    nodo2 *primero;
    nodo2 *ultimo;
}Fila;

nodo2 *iniciLista();
nodo2 *crearNodo2(int dato);
nodo2 *borrarPrimero(nodo2 *lista);
nodo2 *agregarAlFinal(nodo2 *lista, nodo2 *nuevo);
nodo2 *buscarNodo(nodo2 *lista, int dato);

void inicFila(Fila *fila);
void mostrarLista(nodo2 *lista);
void encolar(Fila *fila, nodo2 *nuevo);
void desencolar(Fila *fila); //NO DEVUELVE NADA, SOLO ELIMINA

nodo2 *desencolar2(Fila *fila); //DEVUELVE EL NODO2 QUE SE RETIRA

void mostrarFila(Fila *fila);

int menu();

int main()
{
    Fila fila;
    inicFila(&fila);

    int opcion=0;
    while(opcion!=27)
    {
        opcion=menu();
        switch(opcion)
        {
        case 27: //ESC
            printf("\Saliendo!");
            break;

        case 49:// 1- AGREGAR UN NODO A LA FILA
            printf("\n Ingrese un dato a cargar: ");
            int valor;
            scanf("%d",&valor);
            nodo2 *nuevo=crearNodo2(valor);
            encolar(&fila,nuevo);

            printf("\n Agregado!");
            break;

        case 50:;//2- REMOVER UN ELEMENTO DE LA FILA
            //desencolar(&fila); //DESENCOLA SIN DEVOLVER EL NODO RETIRADO
            nodo2 *desencolado=desencolar2(&fila);
            if(desencolado)
            {
                printf("\n Se elimino el nodo %i \n",desencolado->dato);
            }
```

```
            free(desencolado);
            break;

        case 51:// 3- MOSTRAR LA FILA
            mostrarFila(&fila);
            break;

        default:
            printf("\n Opcion invalida! ");
            break;
        }
    }
    return 0;
}

nodo2 *iniciLista()
{
    return NULL;
}
nodo2 *crearNodo2(int dato)
{
    nodo2 *nuevo=(nodo2*)malloc(sizeof(nodo2));
    nuevo->anterior=NULL;
    nuevo->siguiente=NULL;
    nuevo->dato=dato;

    return nuevo;
}

nodo2 *borrarPrimero(nodo2 *lista)
{
    if(lista)
    {
        nodo2 *seg=lista->siguiente;
        free(lista);
        lista=seg;
    }
    return lista;
}

/*
nodo2 *agregarAlFinal(nodo2 *lista, nodo2 *nuevo)
{
    if(!lista)
    {
        lista=nuevo;
    }
    else
    {
        lista->siguiente=nuevo;
        nuevo->anterior=lista;
    }
    return nuevo;
}
*/

nodo2 *agregarAlFinal(nodo2 *lista, nodo2 *nuevo)
{
    if(!lista)
    {
        lista=nuevo;
    }
    else
    {
        nodo2 *iterador=lista;
        while(iterador->siguiente)
```

```
133         {
134             iterador=iterador->siguiente;
135         }
136 
137         iterador->siguiente=nuevo;
138         nuevo->anterior=iterador;
139     }
140     return lista;
141 }
142 
143 nodo2 *buscarNodo(nodo2 *lista, int dato)
144 {
145     nodo2 *resultado;
146     if(lista)
147     {
148         if(lista->dato==dato)
149         {
150             resultado=lista;
151         }
152         else
153         {
154             nodo2 *iterador=lista;
155             while(iterador && !resultado)
156             {
157                 if(iterador->dato==dato)
158                 {
159                     resultado=iterador;
160                 }
161                 iterador=iterador->siguiente;
162             }
163         }
164     }
165     return resultado;
166 }
167 
168 void inicFila(Fila *fila)
169 {
170     fila->primero=iniciLista();
171     fila->ultimo=iniciLista();
172 }
173 
174 void mostrarLista(nodo2 *lista)
175 {
176     if(lista)
177     {
178         nodo2 *iterador=lista;
179         while(iterador)
180         {
181             printf("         \n %x| %i | %s |%x \n",iterador->anterior,iterador->
dato,iterador->palabra,iterador->siguiente);
182             iterador=iterador->siguiente;
183         }
184     }
185     else
186     {
187         printf("\n la lista esta vacia! ");
188     }
189 }
190 
191 void encolar(Fila *fila, nodo2 *nuevo)
192 {
193     fila->ultimo=agregarAlFinal(fila->ultimo,nuevo);
194     if(fila->primero==NULL)
195     {
196         fila->primero=fila->ultimo;
197     }
```

```
198     fila->ultimo=nuevo;
199 }
200
201 void desencolar(Fila *fila)
202 {
203     if(fila->primero)
204     {
205         if(fila->primero==fila->ultimo)
206         {
207             printf("\n Se eleminara! %i \n",fila->primero->dato);
208             free(fila->primero);
209             fila->primero=NULL;
210             fila->ultimo=NULL;
211         }
212         else
213         {
214             printf("\n Se eleminara! %i \n",fila->primero->dato);
215             fila->primero=borrarPrimero(fila->primero);
216         }
217     }
218 }
219
220 nodo2 *desencolar2(Fila *fila)
221 {
222     nodo2 *desencolado=(nodo2*)malloc(sizeof(nodo2));
223     if(fila->primero)
224     {
225         if(fila->primero==fila->ultimo)
226         {
227             printf("\n Se eleminara! %i \n",fila->primero->dato);
228             *desencolado=*fila->primero; //COPIA EL CONTENIDO DE UN PUNTERO EN OTRO
229             free(fila->primero);         //DADO QUE FREE() HABILITA A QUE EL SECTOR
230             fila->primero=NULL;          //DE MEMORIA PUEDA SER SOBREESCRITO
231             fila->ultimo=NULL;
232         }
233         else
234         {
235             printf("\n Se eleminara! %i \n",fila->primero->dato);
236
237              *desencolado=*fila->primero;
238
239             fila->primero=borrarPrimero(fila->primero);
240
241         }
242     }
243     return desencolado;
244 }
245
246 void mostrarFila(Fila *fila)
247 {
248     if(fila)
249     {
250         mostrarLista(fila->primero);
251     }
252     else
253     {
254         printf("\n la lista esta vacia!");
255     }
256 }
257
258 int menu()
259 {
260     int seleccion;
261     printf("\n 1- Agregar elementos a la fila ");
262     printf("\n 2- Retirar elementos de la fila");
263     printf("\n 3- Mostrar la fila");
```

```
264       printf("\n \n ESC - SALIR");
265
266       seleccion=getch();
267   }
```